SOLUTIONS TO THE PROBLEM

A deadlock avoiding system of the present invention comprises: periodic monitoring means for periodically determining a deadlock with serially reusable resources; a task detection means for detecting a deadlocked task; batch releasing means for comparing execution levels of the detected deadlocked tasks, for releasing all serially reusable resources locked by tasks having low execution levels, and for preferentially running tasks having high execution levels; context saving means for saving contexts when the serially reusable resources locked by tasks having low execution levels are released; context restoring means for restoring the contexts when the tasks having low execution levels are re-started; and batch securing means for collectively re-securing the released serially reusable resources.

⑲日本国特許庁（JP）　⑪特許出願公開

⑫公開特許公報（A）　平4－85632

⑤Int. Cl.⁵	識別記号	庁内整理番号
G 06 F 9/46	340 G	8120－5B
	313 Z	8120－5B
11/30	305 G	7165－5B
12/00	535 A	8944－5B
15/16	470 A	9190－5L

⑬公開　平成4年(1992)3月18日

審査請求　未請求　請求項の数 1　（全5頁）

⑭発明の名称　デッドロック回避方式

⑳特　願　平2－199008

㉒出　願　平2(1990)7月30日

㉑発明者　大野　重幸　東京都港区芝5丁目7番1号　日本電気株式会社内

㉑出願人　日本電気株式会社　東京都港区芝5丁目7番1号

㉑代理人　弁理士　山下　穣平

明　細　書

1．発明の名称

デッドロック回避方式

2．特許請求の範囲

定期的に逐次使用可能資源のデッドロックを判定する定期監視手段と、デッドロック中のタスクを検出するタスク検出手段と、検出されたデッドロック中のタスクの実行レベルを比較し、実行レベルの低いタスクがロックしているすべての逐次使用可能資源を解放し、実行レベルの高いタスクを優先して走行させるための一括解放手段と、実行レベルの低いタスクの逐次使用可能資源解放時のコンテクストを退避するためのコンテクスト退避手段と、実行レベルの低いタスクの再スタート時にコンテクストを回復するためのコンテクスト回復手段と、解放した逐次使用可能資源を一括して再確保するための一括確保手段とを備えていることを特徴とするデッドロック回避方式。

3．発明の詳細な説明

〔産業上の利用分野〕

本発明は、密結合構成のマルチプロセッシングシステムにおける逐次使用可能資源のデッドロック回避方式に関する。

〔従来の技術〕

従来、密結合マルチプロセッシングシステムにおいて、同時使用不可であるが逐次使用の可能な資源、すなわちＳＲＲ（Serially Reusable Resource）のデッドロックが発生した場合には、デッドロックを起こしている双方のタスクの実行レベルを比較して実行レベルが低い方のタスクが確保しているＳＲＲをすべて解放し、実行レベルが低い方のタスクを最初のＳＲＲ確保以前の状態に戻すことによってデッドロックを回避していた。

第4図は上述した従来の方式の具体例を示す図であり、ＣＰＵ５０のタスクＡがＳＲＲ５２を使用するため他タスクからの使用を禁止（ロック）し、またＣＰＵ５１のタスクＢがＳＲＲ５３をロックしているときに、タスクＡがＳＲＲ５３を、タスクＢがＳＲＲ５２をロックしようとした場合に、双方の使用禁止（ロック）を解除できなくな

り、デッドロックとなる。

ここで、タスクＡよりタスクＢの実行レベルの方が低い場合、タスクＢがロックしているＳＲＲ５３を解放し、タスクＢをＳＲＲ５３のロック以前の状態に戻すことによって、デッドロックの回避を行っていた。

〔発明が解決しようとする課題〕

しかしながら、上述した従来の方式では、SRRのデッドロックが発生すると実行レベルが低いタスクは最初にＳＲＲを確保した時点まで戻されてしまうことになり、ＳＲＲのデッドロックが発生した時点までに行った処理が無駄になってしまうという欠点があった。

本発明は、上記のような従来の欠点を改善したもので、その目的は、ＳＲＲのデッドロックを起こしている双方のタスクの実行レベルを比較し、実行レベルの低い方のタスクがロックしている全てのＳＲＲを解放し、実行レベルの低いタスクを再スタートするときにこのタスクがデッドロック発生時までに行った処理が無駄になるのを防止することの可能なデッドロック回避方式を提供することにある。

〔課題を解決するための手段〕

本発明のデッドロック回避方式は、定期的に逐次使用可能資源のデッドロックを判定する定期監視手段と、デッドロック中のタスクを検出するタスク検出手段と、検出されたデッドロック中のタスクの実行レベルを比較し、実行レベルの低いタスクがロックしているすべての逐次使用可能資源を解放し、実行レベルの高いタスクを優先して走行させるための一括解放手段と、実行レベルの低いタスクの逐次使用可能資源解放時のコンテクストを退避するためのコンテクスト退避手段と、実行レベルの低いタスクの再スタート時にコンテクストを回復するためのコンテクスト回復手段と、解放した逐次使用可能資源を一括して再確保するための一括確保手段とを有している

〔作　用〕

逐次使用可能資源（ＳＲＲ）のデッドロックが発生した場合に、実行レベルの低いタスクを一時的にストップさせて、これがロックしているSRRをその間解放し、再スタート時には、一括してＳＲＲを再確保することで、実行レベルの低いタスクがデッドロック発生時までに行った処理が無駄になるのを防止している

〔実施例〕

次に本発明について図面を参照して説明する。

第１図は本発明の一実施例のブロック図である。本実施例は、一定時間毎に起動しデッドロックを判定する定期監視手段１１と、デッドロックと判定された場合、デッドロックを起こしているタスクを検出するタスク検出手段１２と、検出された双方のタスクの実行レベルを比較し、実行レベルの低い方のタスクがロックしているすべてのSRRを一括解放し、実行レベルの高いタスクを優先して走行させるためのＳＲＲ一括解放手段１３と、実行レベルの低い方のタスクのＳＲＲ解放時のコンテクスト（タスクが走行するための環境）を退避するためのタンテクスト退避手段１４、実行レベルの低い方のタスクが走行可能になった際、タスクを再スタートするために、退避してあったコンテクストを回復するためのコンテクスト回復手段１５と、ＳＲＲ一括解放手段１３で解放したすべてのＳＲＲを一括して再確保するためのＳＲＲ一括確保手段１６とで構成されている。

次にこのような構成におけるＳＲＲのデッドロック回避処理動作について説明する。

ＳＲＲのデッドロック回避処理を行うために、第２図に示すように、ＳＲＲ１つづつに対応する情報域ＩＮＦを設ける。情報域ＩＮＦには、デッドロックを判定するためのカウンタと、デッドロックしているタスクを検出するための２つのタスクＩＤ格納域とを設ける。

ＳＲＲをロックした時にロックしたＳＲＲのＳＲＲＩＤ（ＳＲＲ識別子）をタスクコントロールブロックＴＣＢからポイントされるＳＲＲロックリストに設定する。さらに、そのＳＲＲに対応する情報域ＩＮＦの１つのタスクＩＤ格納域にロックしたタスク、例えばＡのタスクＩＤ（タスク識別子）を設定する。既にロックしてあるＳＲＲ

に、他のタスク、例えばBがロックしようとした場合、無限リトライの場合に限り、ある回数以上リトライすれば、そのＳＲＲに対応する情報域ＩＮＦのもう１つのタスクＩＤ格納域にそのタスクＢのタスクＩＤを設定する。

またＳＲＲをアンロックした時には、そのSRRに対応する情報域ＩＮＦをクリアしておく。

定期監視手段１１は定期的に起動されるタスクであり、起動されると、先ず、ロックされているＳＲＲがあるか判定（情報域ＩＮＦがクリアされていればロックされていない）し、無ければ再びウエイトする。ロックされているＳＲＲがあれば、そのＳＲＲに対応する情報域ＩＮＦのカウンタを１アップし、それをデッドロック判定しきい値と比較し、デッドロック判定しきい値より大きければ、そのＳＲＲをデッドロックとみなし、タスク検出手段１２を呼び出す。デッドロック判定しきい値以下なら再びウエイトする。

タスク検出手段１２は、定期監視手段１１でデッドロックとみなされたＳＲＲに対応する情報域ＩＮＦの、タスクＩＤからデッドロックになっているタスクを検出する。

ＳＲＲ一括解放手段１３でそのタスクの実行レベルを比較し、実行レベルの低い方のタスクがロックしているＳＲＲをＳＲＲロックリストより求め、すべて解放する

このとき、コンテクスト退避手段１４によって実行レベルの低いタスクのコンテクストをタスクコントロールブロックＴＣＢからポイントされる領域に退避し、該タスクは、実行待ち状態になる。

その後、実行レベルの高い方のタスクの処理が終了し、実行レベルに低い方のタスクが走行可能になった場合、コンテクスト退避手段１４によって退避されているコンテクストをコンテクスト回復手段１５よって回復し、さらに、ＳＲＲ一括解放手段１３で解放したすべてのＳＲＲをＳＲＲ一括確保手段１６によって確保することによって、実行レベルの低いタスクをデッドロック発生時点から再度走行させることができる。

第３図は本実施例による上述した処理方式の具体例を示す図である。

第３図を参照すると、デッドロックが発生すると、タスクＡよりタスクＢの実行レベルの方が低い場合には、タスクＢがロックしているＳＲＲ53を解放し、タスクＢのＳＲＲ５２のロック失敗時のコンテクストを退避する。タスクＡの処理が終了し、タスクＢが走行可能になった際、ＳＲＲ53を確保し、退避してあったコンテクストを回復することによって、ＳＲＲ５３のロック時からSRR５２のロック時までの間にタスクＢが行った処理が無駄になるのを防ぎタスクＡを優先的に走行させてデッドロックの回避を行うことができる。

〔発明の効果〕

以上説明したように、本発明は、ＳＲＲのデッドロックが発生した場合、実行レベルの低い方のタスクを一時的にストップさせ、そのロックしているＳＲＲをその間解放し、再スタート時に一括してＳＲＲを再確保することによって、実行レベルの低いタスクがデッドロック発生時までに行った処理が無駄になることを防止し、デッドロックを回避することができるという効果がある。

４．図面の簡単な説明

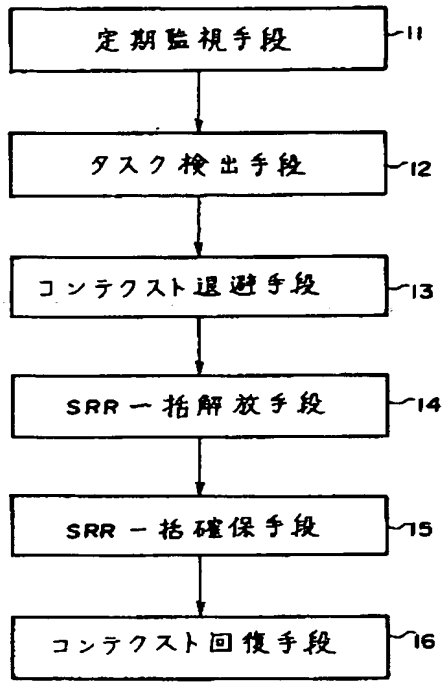
第１図は本発明の一実施例のブロック図、第２図はＳＲＲとタスクとの関連を示す図、第３図は本発明によるデッドロック回避処理の具体例を示す図、第４図は従来のデッドロック回避処理の具体例を示す図である。

第１図において、

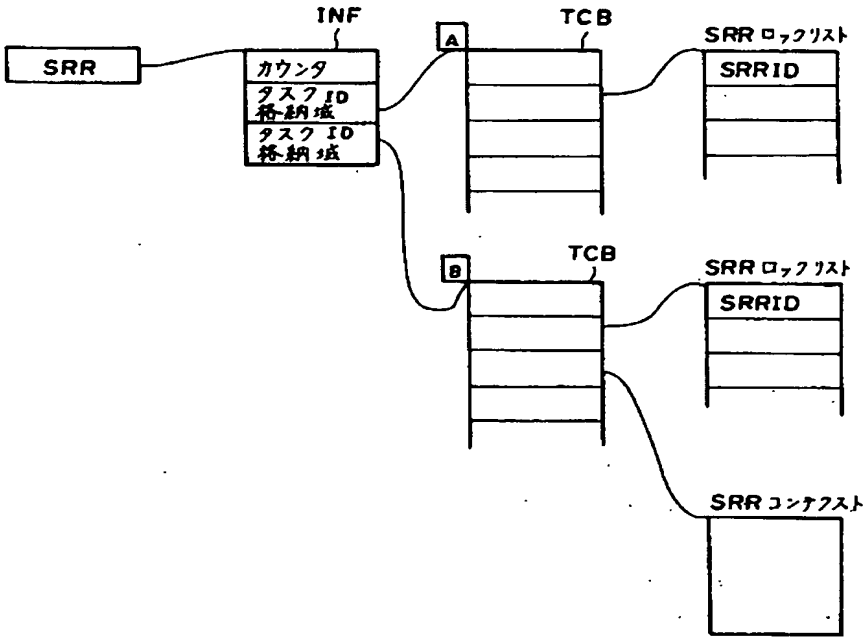
１１…定期監視手段、１２…タスク検出手段、１３…コンテクスト退避手段、１４…ＳＲＲ一括解放手段、１５…ＳＲＲ一括確保手段、１６…コンテクスト回復手段。

代理人　弁理士　山下穣平

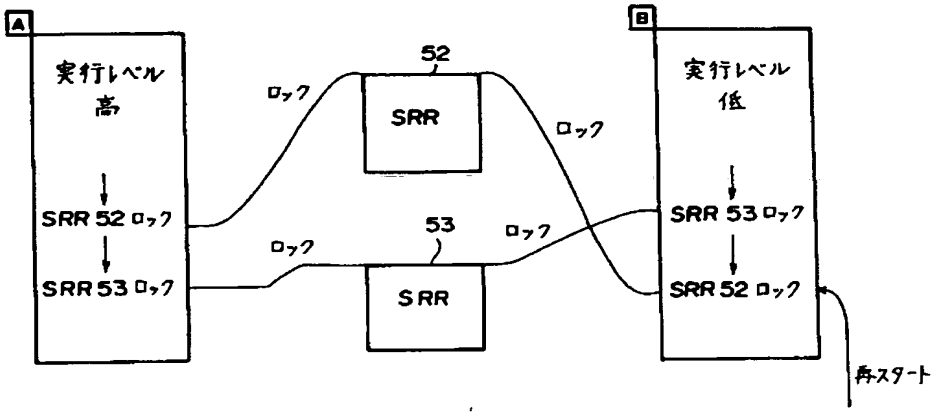
第 1 図


定期監視手段
11
タスク検出手段
12
コンテクスト退避手段
13
SRR一括解放手段
14
SRR一括確保手段
15
コンテクスト回復手段
16

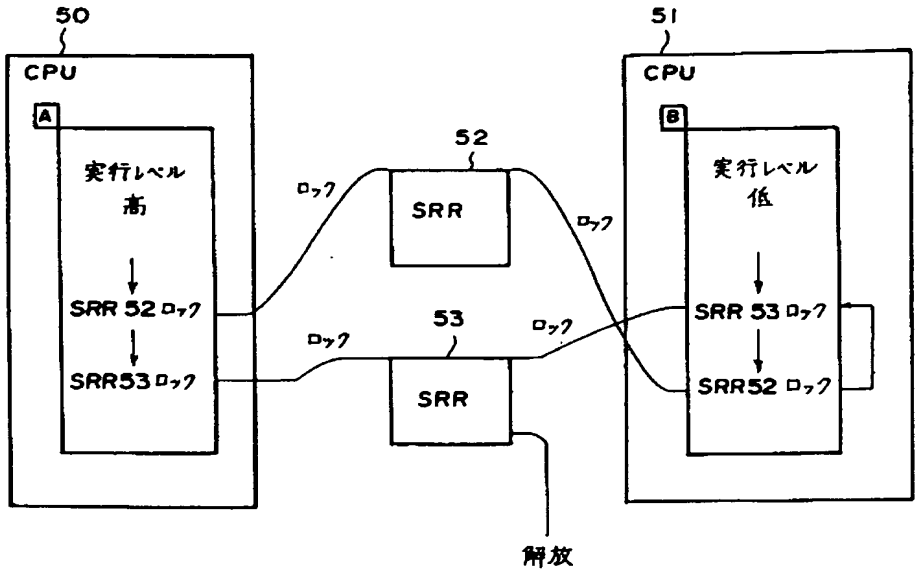
第 2 図


INF
SRR
カウンタ
タスクID格納域
タスクID格納域
A
TCB
SRRロックリスト
SRRID
B
TCB
SRRロックリスト
SRRID
SRRコンテクスト

第 3 図


A
実行レベル
高
SRR 52 ロック
SRR 53 ロック
ロック
52
SRR
ロック
ロック
53
SRR
ロック
B
実行レベル
低
SRR 53 ロック
SRR 52 ロック
再スタート

第 4 図


50
CPU
A
実行レベル
高
SRR 52 ロック
SRR53 ロック
51
CPU
B
実行レベル
低
SRR 53 ロック
SRR52 ロック
52
SRR
53
SRR
ロック
ロック
ロック
ロック
解放